Modeling Intuition as a System

"To explain the integration of information, we need only exhibit mechanisms by which information is brought together and exploited by later processes."   
**-David Chalmers, Facing Up to The Problem of Consciousness**

# Introduction

## Objective

General intelligence of the type we possess exists exclusively, for now, in the domain of the conscious human. Therefore, an understanding of the mechanisms leading to our conscious experience may be required if an artificial general intelligence is to one day be realized.

One defining aspect of human intelligence is our ability to subconsciously form new connections between abstract concepts, which then seem to "bubble up" to the forefront of our attention. This phenomenon, commonly called intuition, is responsible not only for our most startling and profound "Aha!" moments, but also for the seemingly arbitrary changes in our awareness of, say, the ticking of a clock on the wall.

Although intuition is, unfortunately, a system that exists inside us as a “black box” (we have no conscious access to its decision-making process), the ways that we experience these shifts of attention unwillingly and "out of the blue" provide powerful clues to its underlying mechanisms. With that in mind an ensemble learning system (the *agent*) who's awareness (its *attention*) is directed according to some optimization function (an *intuition)* with the goal of recognizing symbols in its search-space (*environment*) and forming new symbolic connections between them.

## Technologies

The framework and agent were developed in the Python (3.6) programming language, chosen for its rapid prototyping capabilities and reflective nature. 3rd party library usage includes KarooGP (genetic programming) and PyTorch (machine learning).

## Problem Domain

To facilitate exploration of this topic, a problem domain was chosen based on the realization that the agent’s environment should, ideally:

1. Afford the agent an opportunity to explore a complex, unknown search-space.
2. Be multi-context.
3. Provide mechanisms for signaling feedback to the agent.
4. Have the potential for practical application.

With that in mind, the agent was applied to the task of learning Python programming language and, eventually, accomplishing some goal using it. This meets our criteria very well, because Python -

1. Is a complex space in which an isolated learning environment may be constructed.
2. Provides various contexts. Ex: keywords, functions, arguments, classes, instances, programs, etc.
3. Exposes methods such as keyword.iskeyword(s) and callable(s), enabling the agent to “ask” if some string s represents a Python keyword or a callable function, respectively. Further, because the agent itself is written in Python, custom methods may be written for the agent to provide more specific feedback, such as is\_python(s) which returns True if s represents a valid Python program with no syntax errors that generates no exceptions.
4. Is highly reflexive, allowing a sufficiently advanced agent to, quite literally, write a version of itself that solves some arbitrary problem. Indeed, even writing its own feedback mechanisms for querying fitness heuristics.

# The Model

## Design Considerations: Modeling Intuition

Subjectively (from the perspective of our awareness) experiencing intuition “feels” no different than experiencing input from any of our garden-variety five senses except for (at least) one notable difference: intuitive input carries with it contextual meaning and symbolic comprehension about our environment - ideas composed by filtering environmental input through the sieve of one’s accumulated life experience. In this way it might be thought of as a sixth sensory organ, different from the first five in that it serves information from our sub-conscience.

Our intuitive model was conceived to represent this “sixth-sense” interpretation of intuition, and according to the following observations of behavior in humans -

* Observation 1  
  a) We are capable of reacting to events faster than we have time to logically determine a rational course of action(CITE). Regardless, we may still make very good “in-the-moment” approximations that seem to involve no conscious thought.  
    
  b) We can articulate the rules we use to solve a given problem but are generally unable to explain why we chose to consider one specific set of rules over another (Pitrat, 2010).  
    
  c) Shifts in changes to our awareness are often (if not always) autonomic (CITE). For example, we cannot dictate when a song will become stuck in our head, or which clock we can suddenly hear ticking.   
    
  \* Conclusion 1  
  Some system operating below our level of consciousness exists for selectively serving information into our awareness.
* Observation 2  
  Seemingly trivial and/or unproductive “mistakes” often come into our awareness. For example, songs DO get stuck in our head and we DO become suddenly aware of a ticking clock for no apparent reason. Contrapositively, we’re often oblivious to important environmental queues, especially when properly distracted (a trait commonly exploited by magicians and pick pockets alike).  
    
  \* Conclusion 2   
  Intuition is not perfect. However, "mistakes” have evolutionary value (e.g. genetic mutation compels biological adaptation). Therefore, as a biological system itself, it is reasonable to assume that the mechanism by which it learns to optimally processes and serve information is Darwinian in nature.
* Observation 3  
  The state of our awareness affects our intuition. Contrarywise, the state of our intuition affects our awareness. To demonstrate this, consider what occurs when focusing one’s awareness on a particular topic; we become acutely aware of *it* and much less aware of everything *else* – apparently dictating either the amount of work performed on, or the priority assigned to, specific information related to the subject of our intense interest.  
    
  \* Conclusion 3  
  Awareness and intuition exist as a feedback loop, each guiding each other in lockstep.
* Observation 4  
  We possess the ability hold, and operate on, a limited set of arbitrary symbols in our head at one time that may used to solve a given problem. Summing three numbers in one’s head, for example.  
    
  Conclusion 4  
  Humans possess a short-term “working memory” that is accessible to the intuition.
* Observation 5  
  It is human nature to thoroughly explore our environment for personal gain. As evidence of this, I offer humankind’s domination of its natural habitat and technological innovation.  
    
  \* Conclusion 5  
  An environmentally-aware agent motivated by evolutionary forces and possessing an intuition may naturally act to explore its environment and actively develop mechanisms for its exploitations.

## The Model

The intuitive model (Fig. 1) consists of 4 layers, labeled ***Classification****,* ***Evolutionary****,* and ***Logical***.   
Data is mostly feed-forward with a single feedback channel for backward-signaling fitness metrics.  
Two output channels also exist: performance metrics (e.g. for logging) and actual agent output.

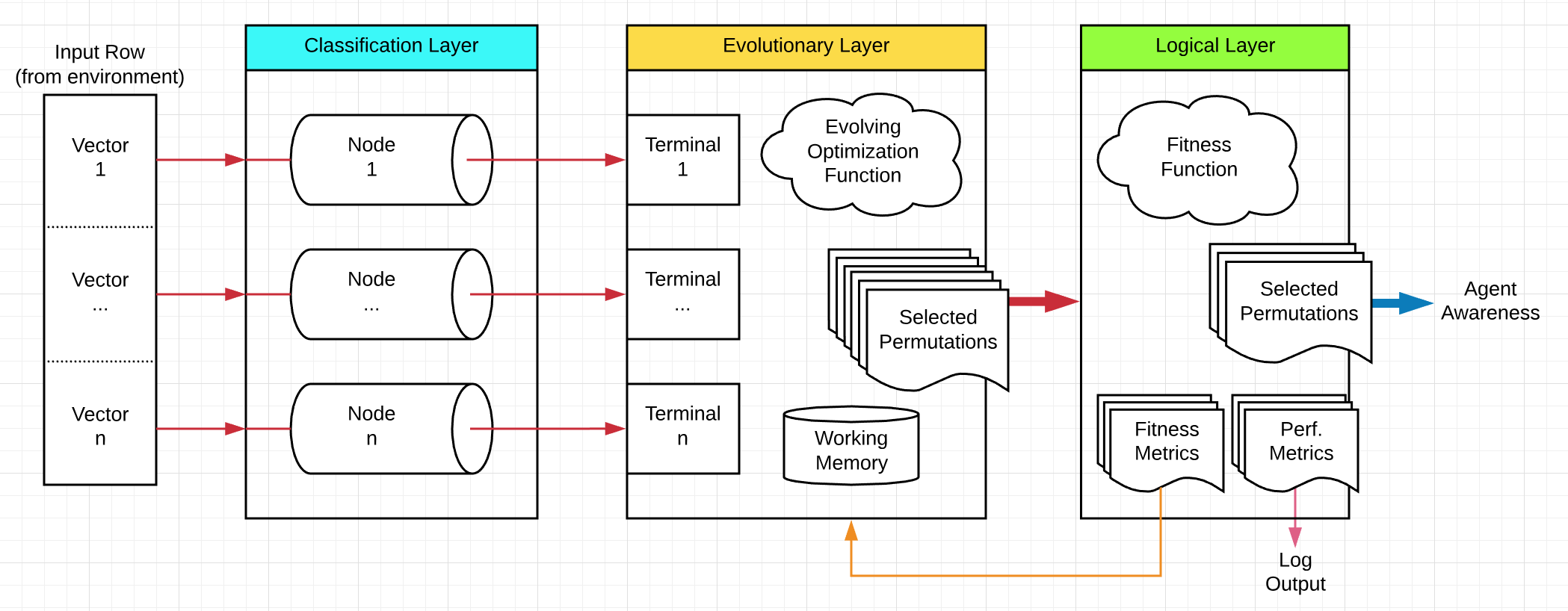


Figure 1

### Layer 1 – The ***Classification*** Layer

**Represents our ability to rapidly (intuitively) classify patterns found in our environment into symbols based on our previous exposure to similar patterns.**

This layer is a set of classifiers built as an arrangement of d parallel **nodes** and can be thought of as an input bus, where each node is a single line on the.

###### Input

Input is received by this layer as a row of environment data. Each row is a collection of *vectors*, one for each node. Thus, input to each node is a sample of some subset of the agent's environment.

(Format note: d = the number of vectors in the input row, a number that should be consistent across all rows of environmental input data. If it is not consistent, the smallest width among all rows is used.

###### Operation

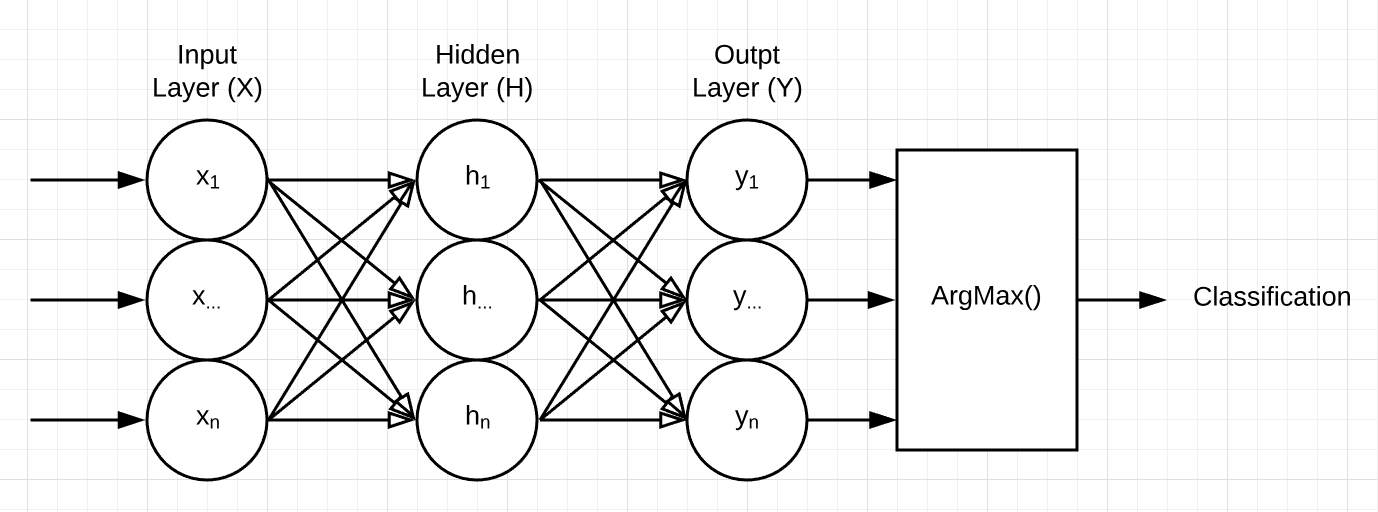
Each node assigns its given input vector a label (a *classification*) from its pool of pre-learned symbol labels.

Learning note: Each node is trained independently (though still in parallel) in a supervised and offline fashion from a training/validation set specific to that that node.

Output   
This layer’s output is the set C of all node classifications for the current set of input row vectors.

###### Implementation

Each classification-layer node is a classifier implemented as an artificial neural network (ANN) of the following shape (Fig. 2). The depth of each ANN’s layers is determined dynamically based on the properties of the training data used to train that node.



|  |  |  |  |
| --- | --- | --- | --- |
|  | **X** | **H** | **Y** |
| **Input Function** | Linear | Linear | Linear |
| **Activation Function** | Rectified Linear | Rectified Linear | Sigmoid |

Figure 2

### Layer 2 – The ***Evolutionary*** Layer

**Represents our ability to form new symbolic connections from existing symbols in our environment.**

This layer is composed of two parts, 1) An optimization function of k output templates, and 2) A short-term working memory of a predefined depth j allowing the layer to generate its output from both its current inputs (called *terminals*) and the previous j inputs.

###### Input

Input is received from the ***classification*** layer as set C (having size d) of symbols.

###### Operation

A string is generated from each output template as an arrangement of symbols from the symbol-pool S. where S is C plus the previous j instances of C.

Learning Note: Each template is optimized in an online manner as fitness is signaled back from the ***logical*** layer. In this way the agent, as it progresses, learns to optimally allocate its "attention" while still allowing "mistakes" to enter its awareness. These mistakes represent possible new conceptual connections and will be validated by the subsequent layer.

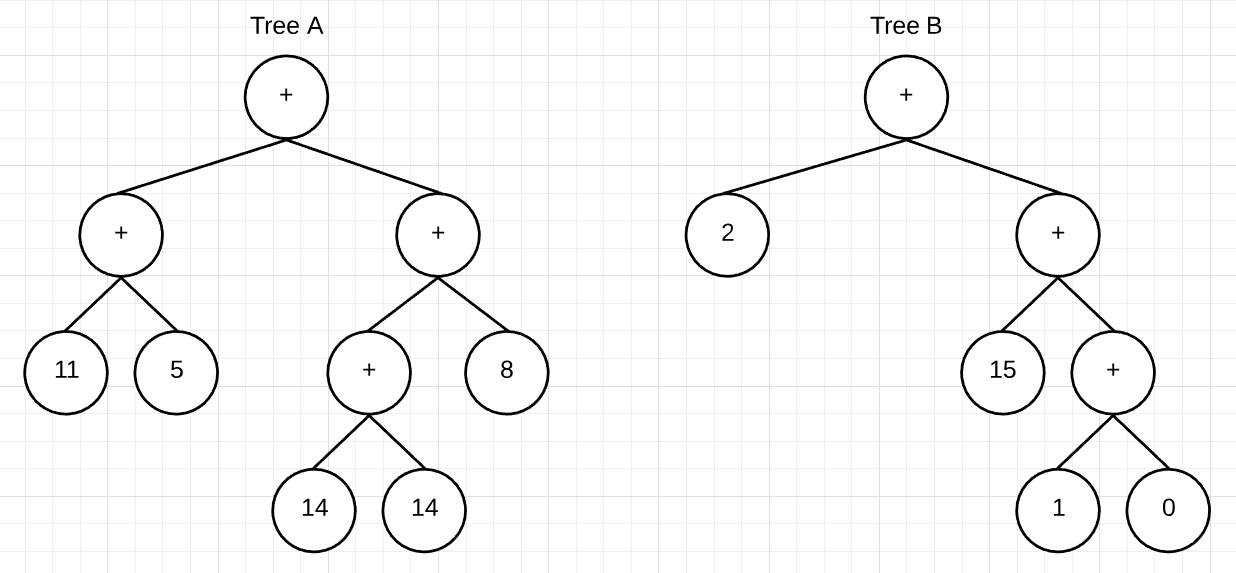
###### Output

This layer’s output is the set P (having size k) of strings.

###### Implementation

The optimization function is represented internally as a set K of k genetically evolving binary expression trees. The leaf nodes of each tree denote a single index in G, thus the sequence of a tree’s leaf nodes denotes a sequence of symbols from G.

For example -

Suppose:  
S = {‘D’, ‘R’, ‘W’, ‘V’, ‘U’, ‘E’, ‘G’, ‘T’, ‘I’, ‘A’, ‘F’, ‘H’, ‘K’, ‘S’, ‘L’, ‘O’}, and   
K = {Tree A, Tree B}, where Tree A and Tree B are given by -  
  
  
Then:  
Tree A denotes the string “HELLO” and Tree B denotes the string “WORLD”.  
  
And therefore:  
Output set P = {“HELLLO”, “WORLD”}

### Layer 3 – The ***Logical*** Layer

**Represents our ability to validate ideas against the environment.**

This layer provides the mechanisms by which the agent validates the contents of its awareness against its environment as well as computing performance metrics such accuracy and processing time.

###### Input

Input is received from the ***evolutionary*** layer as set P of strings.

##### Operation

Each string in P is validated according to the context mode. Fitness information is then sent as feedback to the ***evolutionary*** layer.

###### Output

This layer’s output is a set of strings representing the agents current awareness.  
Initially these strings will appear random, but as the agent progresses they will begin to “narrow in” on the symbolic connections in its environment.

###### Implementation

The context mode defines this layer and is implemented as a function returning either True or False.

For example -

Suppose:  
P = {“HELLO”, “WORLD”}, and   
context mode = is\_noun()

Then:  
fitness = {0, 1}

# The Agent

The agent contains the intuitive model and steps each row of environment input data through it.

Each row sent through the model constitutes a single **step**, and the goal of a step is to 1) classify the current row’s data, 2) tokenize the classifications according to the optimization function, 3) assign each token a fitness value, 4) update the optimization function according to fitness, and 5) serve tokens to the agent.

Each step proceeds, goal-wise, as follows -

1. Input vectors V (segments of the agent’s current input row*)* are fed into the ***classification*** layer.
   1. Each pre-trained classifier node ni assigns each vi a classification label ci.
2. Each ci (a *terminal* *symbol*) is received by the ***evolutionary***-layer’s corresponding terminali.
   1. The current set of terminals are added to working memory. (If working memory now exceeds its predefined depth, the oldest set of terminals in working memory is removed.)
   2. A predefined number of permutations P are generated, each from one or more terminals in working memory according to the optimization function’s current set E of expressions.
3. P is received by the ***logical*** layer.
   1. Each pi is evaluated for fitness according to the agent’s *context mode* (ex: is\_noun(token)). If pi is found to be a productive permutation, the fitness metric for ei is incremented and pi is added to output set O
4. Fitness data for E is signaled back and received by the ***evolutionary*** layer.
   1. A new generation of E is bred from (and replacing) the previous generation in a competitive environment.
5. O is made available to the agent.

## Sample Agent

# Validation Campaign / Results

Upon completion of development, an agent was brought online according to the following campaign –

## Tuneable Parameters

…

## Data Sets

The data set

Data Set desc: <https://archive.ics.uci.edu/ml/datasets/Letter+Recognition>

This data set was chosen for the following reasons

* Familiar with it.
* Deep learnable but w/Simplicity - No need for conv or pooling of layers (though agent is capable of doing so).

## Training the Classification Layer

….

## Sample Run

…

# Performance

* Human context switching is approx 200 ms. 20-50ms is reasonably real-time.

# Failed Approaches/Models

…

# The Framework

Along the way, a framework sprang up…

Framework Note: Many sub-layer components can do their own logging. However, when instantiated from the agent they instead pass their log statements up for handling there.

# What’s Next

Bootstrapping:

* Each sub-agent represents a single context

Branching/Bootstrapping:

1. Branch into a hierarichal structure of agents (faciliated by reflection) – each representing one context. Contexts are not necessarily predefined.
2. L2 branch after some number of increases of the last x iters
3. Each agent represents a single "concept" such as a letter, or a python kwd
4. If a branch agent does not learn enough over some t, rm it (log to kb?) - it's inputs do not form any concepts
5. One node has one goal, a clique has another goal, a network has an even still more encompassing goal (ex: find free memory?)

L2 Tuning:

1. Monitor accuracy heuristics over time -
2. increase max pop size after some accuracy threshold
3. decrease max pop size if proprtion of unfit to fit outputs calls for it

More heuristics:

* What heuristics might drive it and its exploration - neophilia? Self-actualization? Guilt?
* We encounter many heuristics in life
* OBSERVATION: We are not always compelled to act rationally - in addition to logical reasoning, emotional (irrational) reasoning strongly influences our behavior.   
    
  The heuristics guiding our intuition’s evolutionary journey may be associated with irrational drivers such as guilt, loneliness, boredom, etc.
  + Innate drive based on environmental queues/heuristics comples us toward the new, and
  + How are the inputs wighted? Biased by "amount" of input and log(type)? I.e. Fire vs. TV - Fire is big and new.
  + What drives the shifts?
  + The genetic alg decides attentive allocation between input data elements. Heuristics?
* In addition, noting the hierarchal nature of information, the agent was designed to scale from a single agent, to a node-agent in a network of such agents, thereby facilitating the bootstrapping of an increasingly advanced intuition.

# Appendix A:

## Influences

Nicolai Tesla, Daniel Hofstadter, Ray Kurzweil, Richard Dawkins, Jacques Pitrat, Danial Kahneman

# References

Pitrat, J. (2010). Artificial beings Wiley-ISTE.

# Appendix B: Technical Information

## Usage Instructions

…

CLI –

Tunable constants –

## Layer Implementations